Framework Professor-Aluno:
Uma Abordagem de Aprendizado por Reforço

Eduardo Vieira e Sousa
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
eduardoatrcomp@gmail.com

***Abstract*—Este artigo aborda a implementação do Framework Professor-Aluno e investiga seus efeitos sobre o treinamento de um agente para resolver o jogo CartPole. Ambos os agentes, professor e aluno, utilizam-se do algoritmo de aprendizado por reforço conhecido como Q-Learning, combinado a uma rede neural profunda, uma abordagem conhecida como DQN. Dos algoritmos presentes no framework, três foram investigados, sendo eles: EarlyAdvising, ImportanceAdvising e MistakeCorrecting. Esses métodos foram utilizados para treinar um novo agente, e comparados ao treinamento de um agente sem a utilização do framework. Por fim, algumas variações de parâmetros do framework também foram investigadas, e todos os resultados foram analisados.**

I. Introdução

A área do aprendizado por reforço **(Reinforcement Learning)** vem se consolidando nos últimos anos de pesquisa, e alcançando, uma após a outra, inquestionáveis conquistas. No entanto, a comparação entre o aprendizado de máquina e a forma como os seres humanos adquirem conhecimento torna-se quase que inevitável, sendo que é possível perceber uma grande diferença entre os dois, especialmente no que diz respeito à eficiência com a qual são utilizadas as informações obtidas.

Seres humanos não precisam readquirir todo o conhecimento do zero sempre que realizam uma tarefa diferente, porém isso é o que as máquinas estão fazendo no momento. Algumas abordagens para aumentar a eficiência na utilização das informações apresentam técnicas onde é possível acelerar o processo de treinamento de um agente, isso pode ser feito demonstrando soluções bem-sucedidas de uma tarefa. Esse tipo de aprendizado é conhecido como aprendizado por imitação ou demonstração **(Imitation Learning)** [2]. Existem também abordagens que tentam utilizar o conhecimento já adquirido em tarefas similares, esse processo é conhecido como **Transfer Learning** [3].

Deste modo, baseando-se na ideia de que informação extra pode ser explorada na tentativa de acelerar o processo de aprendizado, da mesma maneira que nos seres humanos, foram propostas abordagens onde um agente artificial seria encarregado de “ensinar” outro a executar uma determinada tarefa. Esse problema foi primeiramente estudado por Torrey e Taylor [1], que introduziu um framework de aprendizado por reforço conhecido como o “Framework Professor-Aluno”, onde um conjunto de heurísticas determina as situações onde um agente já treinado para resolver uma certa tarefa, também chamado de "Professor", indica a um agente em treinamento, que faz o papel de "Aluno", as ações corretas a serem tomadas, o que acelera o processo de treinamento do agente.

II. Teaching on a Budget

No framework para aprendizado por reforço introduzido por Torrey e Taylor [1], o agente referido como aluno, aprende a realização de uma tarefa, enquanto um outro agente, conhecido como professor, tem a possibilidade de sugerir uma ação para cada estado encontrado pelo aluno. A dinâmica do treinamento está expressa na figura (1).

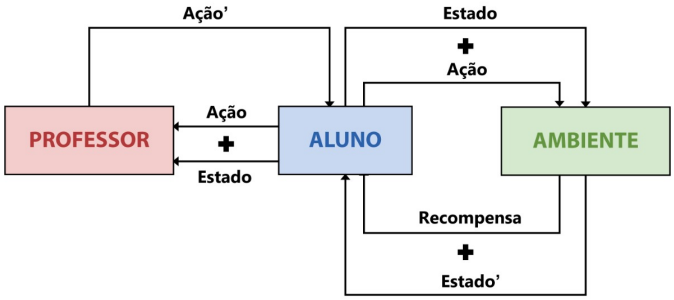

Fig. 1. Dinâmica de treinamento do aprendizado por reforço utilizando o Framework Professor-Aluno

Para que o framework possa ser aplicado, existem algumas condições necessárias. A primeira é a de que o professor possua uma política capaz de resolver a tarefa em questão, e que ambos, aluno e professor, possuam um conjunto de ações em comum, porém é importante ressaltar que o professor não possui acesso aos processos internos do aluno. Deste modo, o framework utiliza uma interface mínima de comunicação entre os agentes, fazendo com que seja possível que cada agente utilize diferentes algoritmos de aprendizado e representações do mundo.

Além disso o conceito de *budget* é introduzido, para que o número de ações sugeridas pelo professor possa ser limitado. Para cada conselho dado pelo agente, uma unidade é subtraída de seu *budget*. Isso se faz necessário, dado que a transmissão de informação no mundo real não é ilimitada, e esse conceito traz consigo a ideia de que o *budget* deve ser administrado da melhor maneira possível, buscando um incremento máximo na performance do aprendizado do aluno, utilizando a menor

quantidade possível de conselhos. Por fim, são apresentados alguns algoritmos que estabelecem a maneira como o *budget* do professor é gasto.

A. Early Advising

A abordagem inicial parte do princípio que o aluno deve se beneficiar mais dos conselhos nos estágios iniciais do aprendizado, quando ele ainda possui pouca informação sobre o ambiente. Deste modo, o **Early Advising** consiste em prover ajuda para todos os primeiros estados, até que o *budget* acabe, como mostra o algoritmo a seguir:

Algorithm 1 EarlyAdvising(π,n)

```
for cada estado s do aluno do
  if n > 0 then
    n ← n − 1
    Advise π(s)
  end if
end for
```

Das abordagens sugeridas por Torrey e Taylor [1], essa é a mais simples, sendo considerada a estratégia base para nossa análise.

B. Importance Advising

Em problemas onde todos os estados possuem a mesma importância, o **Early Advising** pode ser uma boa estratégia. No entanto, isso não é o que ocorre na maioria dos casos, como por exemplo em um jogo de xadrez onde um certo estado pode levar à vitória ou talvez à derrota.

Deste modo, a abordagem **Importance Advising** consiste em guardar o *budget* para que este seja gasto nos momentos mais oportunos, ou seja, em estados com maior importância, que é calculada com através da fórmula (1).

$$I(s) = maxQ(s,a) - minQ(s,a) \tag{1}$$

Deste modo o professor verifica a importância do estado no qual o aluno se encontra, utilizando como referência sua própria função Q. Caso a importância ultrapasse o valor de *threshold* (t), o professor então dá o conselho para o aluno. Caso contrário, o aluno segue tomando as ações de acordo com seu próprio método do aprendizado, como demonstra o algoritmo a seguir:

Algorithm 2 ImportanceAdvising(π,n,t)

```
for cada estado s do aluno do
  if n > 0 and I(s) ≥ t then
    n ← n − 1
    Advise π(s)
  end if
end for
```

C. Mistake Correcting

Mesmo quando o professor guarda o *budget* para os estados mais importantes, ainda é possível que ele esteja sendo desperdiçado. Isso ocorre quando o aluno já possuía a intenção de realizar a ação sugerida, de modo que não seria necessário a intervenção do professor.

Deste modo, o **Mistake Correcting** consiste em analisar a ação tomada pelo aluno, e se esta for diferente da ação sugerida pelo professor, então, da mesma maneira que no **Importance Advising**, o professor analisa a importância do estado e decide se dará um conselho, baseando-se no valor do *threshold* escolhido. O algoritmo está demonstrado a seguir:

Algorithm 3 ImportanceAdvising(π,n,t,a)

```
for cada estado s do aluno do
  if n > 0 and I(s) ≥ t and a ≠ π(s) then
    n ← n − 1
    Advise π(s)
  end if
end for
```

No entanto, é importante ressaltar que para o funcionamento dessa abordagem, é necessário que o professor tenha acesso de alguma maneira a ação que o aluno pretende tomar. Isso pode ser obtido de maneira simples em algoritmos de aprendizado por reforço, sem que o professor tenha conhecimento dos processos internos do aluno, apenas fazendo com que o mesmo anuncie a ação pretendida antes de sua execução.

III. Metodologia

Para a análise do comportamento do Framework Professor-Aluno em uma Deep Q-Network **(DQN)**, foi utilizado o ambiente **CartPole-v1**, implementado na biblioteca **Gym** [7]. Já a rede neural que realiza a tarefa de aproximação de função no algoritmo, foi implementada utilizado a biblioteca **Keras** [8], combinada ao **TensorFlow** [9].

A. Ambiente

O **CartPole-v1** é um ambiente simples que consiste de um mastro localizado sobre um carrinho móvel. A cada turno, o sistema aplica uma força sobre o mastro com o intuito de desequilibrá-lo. O agente deve então tentar reequilibrar o mastro movendo o carrinho para direita ou para esquerda.

A cada turno que o mastro passa em pé, o jogador recebe 1 ponto, e a recompensa máxima para vencer uma rodada é de 500 pontos. No entanto, o episódio acaba caso o mastro se mova mais que 15 graus, para qualquer lado, ou caso o carrinho se mova mais do que 2.4 unidades do centro do mapa. O estado do jogo é representado por quatro valores, como mostra a tabela (I). Já as ações possíveis estão expressas na tabela (II).

Deste modo, um agente capaz de vencer o jogo deve escolher entre empurrar o carrinho para direita ou para esquerda, de maneira correta a fim de equilibrar o mastro. Após 500 turnos de equilíbrio bem sucedidos o agente vence o jogo. Para acelerar o aprendizado, a implementação do framework

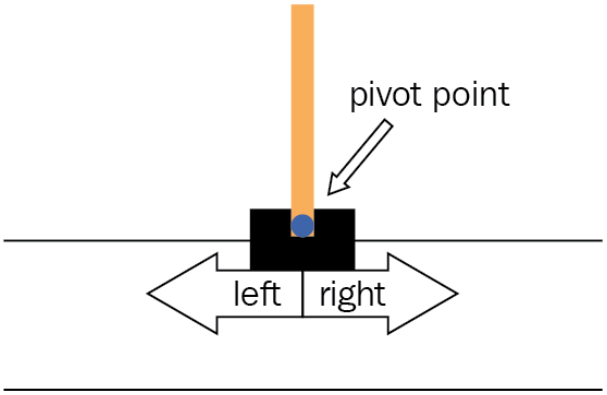

Fig. 2. Renderização gerada pela biblioteca Gym

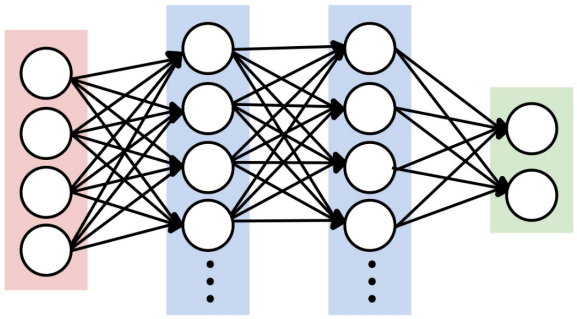

Fig. 3. Arquitetura da rede utilizada

TABLE I
ESTRUTURA QUE REPRESENTA UM ESTADO DO AMBIENTE **CARTPOLE-V1**

Num	Observação	Min	Max
0	Posição do carro	-2.4	$+2.4$
1	Velocidade do carro	$-infinito$	$+infinito$
2	Ângulo do mastro	-41.8	$+41.8$
3	Velocidade na ponta do mastro	$-infinito$	$+infinito$

TABLE II
AÇÕES POSSÍVEIS NO AMBIENTE **CARTPOLE-V1**

Num	Ação
0	Move o carro para esquerda
1	Move o carro para direita

adiciona uma recompensa de -100, caso o agente adote uma ação que o faça perder o jogo.

B. Deep Q-Network

A **DQN** utilizada como aproximador de função para o algoritmo de aprendizado por reforço possui a seguinte arquitetura:

- Camada de Entrada: possui 4 neurônios, cada um recebe um valor que é utilizado na representação de cada estado, como explicado na sessão anterior;
- Camadas Escondidas: a rede possui duas camadas escondidas com 24 neurônios cada. Essa arquitetura possibilita a aproximação de funções não lineares, o que é necessário na solução do ambiente escolhido;
- Camada de Saída: possui 2 neurônios, onde cada um representa uma ação possível para o ambiente, ou seja, mover para esquerda o para a direita.

Deste modo, a arquitetura da rede pode ser representada de acordo com a figura (3) .

Após a escolha da arquitetura da rede, é necessária a definição dos hiper-parâmetros, tanto para o algoritmo de aprendizado como para a rede em si. Deste modo, após uma bateria de testes, foi possível chegar a seguinte combinação de parâmetros

- Fator de Desconto (γ): 0.99
- Taxa de Aprendizagem (α): 0.001
- Fator de Exploração (ϵ): 0.1
- Memory-Replay
  - Tamanho da Memória: 2000
  - Minibatch: 64

Esses valores foram fixados para os hiper-parâmetros durante todos os demais testes. Essa decisão foi tomada com o intuito de observar apenas o efeito do Framework Professo-Aluno, sobre uma rede que converge e é capaz de resolver o problema do **CarPole-v1**.

IV. EXPERIMENTOS E RESULTADOS

Após a implementação de cada algoritmo e o treinamento do professor, antes que seja possível realizar a comparação entre cada abordagem, é necessário definir o valor ideal de *threshold* (T) para o **Importance Advising** e **Mistake Correcting**. Para isso, o aluno foi treinado usando diferentes valores para T, como mostra o gráfico (4) a seguir.

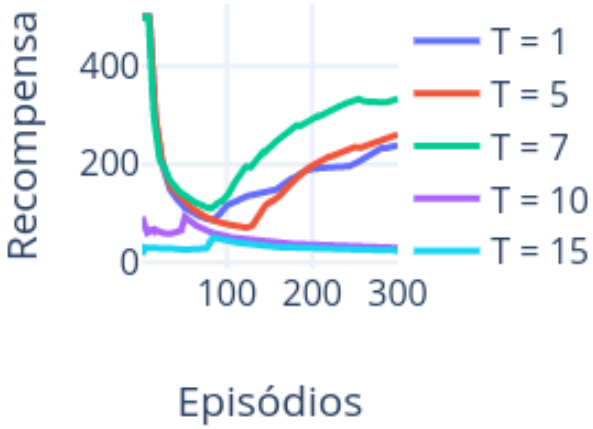

Fig. 4. Testes para os valores de T

Analisando o gráfico é possível perceber que o valor mais eficiente encontrado para T é por volta de 7. O que já pode adiantar que ambos os algoritmos são mais eficientes do que o **Early Advising**, já que esse poderia ser interpretado como um

Importance Advising com um *threshold* de 0. Além disso, também é possível perceber que valores muito altos para o *threshold*, interferem na estabilidade da rede, e no caso do **CartPole-v1**, valores a partir de T=10 fizeram com que a rede não convergisse.

Com os hiper-parâmetros definidos, é possível então iniciar os testes de comparação para cada algoritmo. Os resultados podem ser observados no gráfico (5).

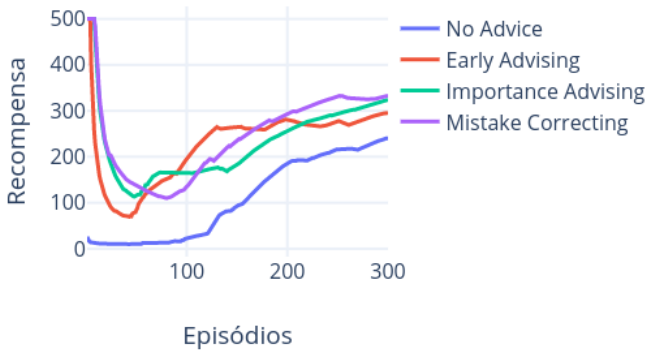

Fig. 5. Recompensa média para cada método abordado

Da mesma maneira que nos trabalhos originais, todos os métodos ocasionaram em uma melhora expressiva da recompensa média, quando comparados a ao treinamento de um agente sem a utilização do framework. O método que mais se destacou foi o **Mistake Correcting**, como era de se esperar, já que é o método que utilizar de maneira mais eficiente o *threshold* fornecido. Em seguida temos o, **Importance Advising** e o **Early Advising**.

Além disso, é possível notar uma crescimento da recompensa média muito mais rápido nos primeiros 100 episódios do **Early Advising**, isso ocorre devido ao fato de que o agente acaba sendo bem treinado para os estados iniciais do game, mas a medida que o treinamento progride, a vantagem de se treinar os estados mais importantes começa a aparecer, como é possível observar nos outros dois algoritmos.

Por fim, os algoritmos também influenciaram bastante na quantidade média de episódios necessários para a convergência do agente, como mostra a tabela (III) a seguir:

TABLE III
QUANTIDADE MÉDIA DE EPISÓDIOS PARA A CONVERGÊNCIA DO MÉTODO

Abordagem	Episódios
No Advice	167
Early Advising	143
Importance Advising	121
Mistake Correcting	93

Em sessões de treinamento com 300 episódios cada, novamente a efetividade de cada algoritmo se manteve. O resultado é de certa maneira bastante intuitivo, visto que uma convergência mais rápida do método, faz com que o agente atinja limite de 500 pontos mais cedo, resultando em uma maior recompensa média até o fim do treinamento.

V. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Os experimentos guiados tiveram o intuito de avaliar apenas o efeito do Framework Professor-Aluno sobre o treinamento do agente. Deste modo, os parâmetros da rede se mantiveram fixos, aos utilizados para o treinamento do professor. No entanto, é bastante plausível que a variação dos parâmetros da rede possa não apenas demonstrar que o framework pode ser ainda mais eficiente, como também podem causar problemas de instabilidade na rede, e fazer com que os métodos não consigam convergir.

Vista a complexidade ocasionada pela quantidade de combinações de métodos e parâmetros possíveis utilizando-se do Framework Professor-Aluno, é natural imaginar que a área ainda necessita de muito experimentação e estudo.

Algumas abordagens interessantes já foram propostas em alguns artigos, como por exemplo um sistema multi-alunos e multi-professores [1], onde poderia ser constatado se o aumento da quantidade de professores ou a interação entre alunos, utilizando diferentes métodos de aprendizado, pode ser benéfica para o processo.

Além disso, também são estudadas técnicas que visam aprimorar ainda mais a utilização do *budget* do professor como novos algoritmos [4], ou também a utilização do conselho por parte do aluno, por meio de uma técnica parecida com o **Experience Replay**, aplicada na **DQN**. Essa técnica consiste em guardar os conselhos dados pelo professor, associados aos seus respectivos estados, e utilizá-los em estados iguais ou similares, essa técnica foi batizada de **Advice-Replay** [5].

Por fim, uma das abordagens mais interessantes é o **Peer-to-Peer Advising** [6], que aplica a abordagem do Framework Professor-Aluno á um sistema multiagente, de forma que cada agente é simultaneamente Professor e Aluno de todos os demais. A ideia é utilizar o conhecimento obtido por cada agente individualmente, de maneira complementar ao conhecimento obtido pelos demais agentes, fazendo assim com que o treinamento seja ainda mais rápido e eficiente.

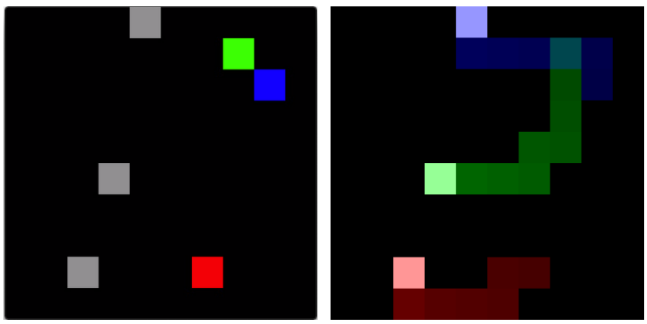

Fig. 6. Imagem do ambiente implementado para o teste do **Peer-to-Peer Advising** em [6]

A figura (6) mostra o ambiente desenvolvido para o teste do **Peer-to-Peer Advising**, onde três agentes são posicionados aleatoriamente, e devem explorar a mapa e se posicionar em cima das áreas marcadas em cinza. Os agentes utilizam o

Framework Profess-Aluno para se ajudar multuamente, econtrando de maneira extremamente rápida as posições marcadas no mapa.

REFERENCES

[1] Torrey, L., Taylor, M.E. (2013). Teaching on a budget: agents advising agents in reinforcement learning. AAMAS.

[2] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. Robot. Auton. Syst.57, 5 (May 2009), 469-483.

[3] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. J. Mach. Learn. Res. 10 (December 2009), 1633-1685.

[4] Zimmer, Matthieu, Viappiani, Paolo, Weng, Paul. (2014). Teacher-Student Framework: A Reinforcement Learning Approach.

[5] Gupta, Vaibhav, Anand, Daksh, Paruchuri, Praveen, Ravindran, Balaraman. (2019). Advice Replay Approach for Richer Knowledge Transfer in Teacher Student Framework.

[6] Ilhan, Ercüment et al. "Teaching on a Budget in Multi-Agent Deep Reinforcement Learning." ArXiv abs/1905.01357 (2019): n. pag.

[7] https://gym.openai.com/

[8] https://keras.io/

[9] https://www.tensorflow.org/